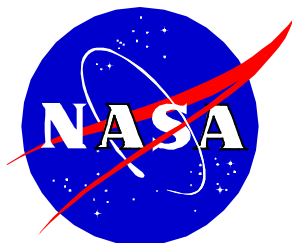


Insert Program Name  
**SOFTWARE MANAGEMENT PLAN**

**Month day, Year**

Insert  
Program  
Logo

**Prepared for**  
**NASA Goddard Space Flight Center**  
**Flight Software Branch/Code 582**  
**Greenbelt, Maryland**



<b>Insert Program Name</b>
<b>SOFTWARE MANAGEMENT PLAN</b>

**Approved by:**

_____ Name Title	_____ Date
------------------------	---------------

_____ Name Title	_____ Date
------------------------	---------------

_____ Name Title	_____ Date
------------------------	---------------

_____ Name Title	_____ Date
------------------------	---------------

## REVISION CHART

List of Affected Pages			
Page Number	CR #	Issue	Publication Date

# TABLE OF CONTENTS

Revision Chart.....	1-3
Table of Contents .....	1-4
List of Tables.....	1-5
1 Introduction.....	1-6
1.1 Project Summary .....	1-6
1.2 Purpose and Scope .....	1-6
1.3 Document Status and Evolution.....	1-6
1.4 Document Organization .....	1-6
2 Related Documentation.....	2-7
2.1 Parent Documents .....	2-7
2.2 Applicable Documents .....	2-7
2.3 Information Documents .....	2-7
3 Project Software Development Overview.....	3-8
3.1 Project Software Development Goals .....	3-8
3.2 System Overview .....	3-8
3.2.1 The Spacecraft.....	3-8
3.2.1.1 Component 1.....	3-8
3.2.1.2 Component 2.....	3-8
3.2.1.3 Component 3.....	3-8
3.2.2 Ground System.....	3-8
3.2.3 Development, Test, and Sustaining Engineering Facilities .....	3-8
3.2.4 Launch Vehicle .....	3-8
3.3 Project Features.....	3-9
3.3.1 Project Phases.....	3-9
3.3.2 Major Milestones .....	3-9
3.3.3 Engineering Teams.....	3-9
3.3.4 Concurrent Engineering .....	3-9
3.4 Project Organization.....	3-9
3.5 Information Infrastructure.....	3-9
4 Management Approach .....	4-1
4.1 Management Objectives and Priorities .....	4-1
4.2 Software Management Organization.....	4-1
4.2.1 Software Management Responsibilities.....	4-1
4.2.2 Software Operations Working Group .....	4-2
4.2.3 Provider Organizational Requirements.....	4-2
4.3 Planning, Monitoring and Controlling Mechanisms .....	4-2
4.3.1 Planning and Estimating .....	4-2
4.3.1.1 Work Breakdown Structure .....	4-3
4.3.1.2 Activity Identification.....	4-4
4.3.1.3 Resource Requirements .....	4-4
4.3.1.4 Assumptions, Dependencies, and Constraints .....	4-4
4.3.1.5 Budget and Resource Allocation .....	4-5
4.3.1.6 Schedules .....	4-5
4.3.2 Progress Assessment .....	4-5
4.3.3 Reviews .....	4-5
4.3.4 Resource and Schedule Management.....	4-6
4.4 Risk Management.....	4-6
4.5 Configuration Management .....	4-6
4.6 Documentation Requirements.....	4-7
4.7 Deviation Procedures .....	4-8
5 Technical Approach .....	5-1
5.1 Software Engineering Process.....	5-1
5.1.1 Life Cycle Model .....	5-1
5.1.2 Formulation Phase.....	5-2

5.1.2.1	Operations Concept Definition .....	5-2
5.1.2.2	Requirements Definition.....	5-2
5.1.2.3	Requirements Analysis .....	5-3
5.1.2.4	Architecture Design .....	5-3
5.1.3	Implementation Phase .....	5-4
5.1.3.1	Software Preliminary Design.....	5-4
5.1.3.2	Software Detailed Design .....	5-5
5.1.3.3	Software Code and Unit Test.....	5-5
5.1.3.4	Software Integration and Test.....	5-5
5.1.4	Operations and Maintenance Phase .....	5-6
5.1.4.1	Deployment.....	5-6
5.1.4.2	Acceptance Testing.....	5-6
5.1.4.3	Transition.....	5-6
5.1.4.4	Operations .....	5-6
5.1.4.5	Sustaining Engineering .....	5-7
5.1.5	Other Software Engineering Considerations.....	5-7
5.1.5.1	Software Categorization and Classification Policy .....	5-7
5.1.5.2	Software Development Folders .....	5-8
5.1.5.3	Data Generation and Management .....	5-8
5.1.5.4	Problem Reporting.....	5-9
5.2	Data Product and Specification Structure .....	5-9
5.3	Technical Performance Measures .....	5-11
6	Product Assurance.....	6-1
6.1	Quality Assurance .....	6-1
6.2	Applicable Standards .....	6-1
6.3	Verification and Validation.....	6-2
6.4	Safety Assurance.....	6-2
6.5	Security .....	6-2
6.6	Certification.....	6-2
7	Abbreviations and Acronyms.....	7-1

## LIST OF TABLES

Table 3-1	Roles of Principle Organizational Elements (sample).....	3-10
Table 5-1	Software Classification.....	5-7
Table 5-2	Project Software Specifications and Data Products.....	5-10

# 1 INTRODUCTION

Insert text

## 1.1 Project Summary

Insert text

## 1.2 Purpose and Scope

Insert text

## 1.3 Document Status and Evolution

Insert text

## 1.4 Document Organization

Insert text

## **2 RELATED DOCUMENTATION**

### **2.1 Parent Documents**

Insert text

### **2.2 Applicable Documents**

Insert text

### **2.3 Information Documents**

Insert text

## 3 PROJECT SOFTWARE DEVELOPMENT OVERVIEW

Insert text

### 3.1 Project Software Development Goals

Insert text

### 3.2 System Overview

Insert text

#### 3.2.1 The Spacecraft

Insert text

##### 3.2.1.1 *Component 1*

Insert text

##### 3.2.1.2 *Component 2*

Insert text

##### 3.2.1.3 *Component 3*

Insert text

#### 3.2.2 Ground System

Insert text

Sample list:

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5

#### 3.2.3 Development, Test, and Sustaining Engineering Facilities

Insert text

#### 3.2.4 Launch Vehicle

Insert text

### **3.3 Project Features**

Insert text

#### **3.3.1 Project Phases**

Insert text

A second designation of project phases corresponds to the traditional NASA system development model with phases named A through E defined as follows.

Phase A –	Preliminary Analysis
Phase B –	Definition
Phase C –	Design
Phase D –	Development
Phase E –	Operations

#### **3.3.2 Major Milestones**

High-level schedules showing the major Project milestones may be found on the Project web site at <http://project.gsfc.nasa.gov>.

#### **3.3.3 Engineering Teams**

Several different teams are presently working on various aspect of the Project. A description of these teams may be found on the Project web site at <http://project.gsfc.nasa.gov>.

#### **3.3.4 Concurrent Engineering**

Insert text

### **3.4 Project Organization**

The Project's organization structure may be found at <http://Project.gsfc.nasa.gov> under "Project Office" and "Who's who".

Insert text

### **3.5 Information Infrastructure**

Insert text

**Table 3-1 Roles of Principle Organizational Elements (sample)**

Project (GSFC Code 4XX)	Systems Engineering Lead SOWG Component 1 Management Component 2 Management Component 2 Space Frame Science Oversight Science Management International Coordination
Mission Project or PI Structure	Ground System Development Mission Operations Science Operations Science Community and Education Outreach
Electrical Systems Center (GSFC Code 5XX)	Flight Data Systems Hardware
Information Systems Center (GSFC Code 5XX)	Flight Software Management Common C&DH Development Component 1 Software Development Test Tools & Facilities Development
Prime Contractor	Spacecraft Development <ul style="list-style-type: none"> <li>• Spacecraft Modules</li> <li>• Component 1</li> <li>• Component 2</li> <li>• Integration &amp; Test</li> <li>• Systems Modeling &amp; Test bed</li> </ul>
Science Community	Instrument Definition Science Advocacy Education & Public Outreach Data Analysis
International Partners	Instrument Definition Spacecraft/Instrument Contributions
Science Development Contractors	Science Instrument Development

## **4 MANAGEMENT APPROACH**

This section describes the methods and processes the Project will use to manage the business and technical aspects of its software development process. It specifies how the Project will plan the work and estimate costs, assess progress, and review products. It describes the principal software management roles and responsibilities and identifies requirements to be levied on software providers to develop and submit business related information needed to accomplish project objectives.

### **4.1 Management Objectives and Priorities**

The Project's approach to managing the development of Project software is intended to ensure, in priority order, (a) that functional, performance and quality requirements are satisfied, (b) that resources are used effectively across the entire life cycle, and (c) that delivery schedules are met.

Pursuant to these objectives, the project will:

- Create and maintain a software management organization that embodies and exploits the project's priority for open communication and information sharing.
- Establish appropriate management methods and practices.
- Maintain awareness of status and progress against schedules and budget.
- Identify and manage areas of risk.
- Establish formal methods and tools for tracking system components and configurations.

### **4.2 Software Management Organization**

The Project software management organization will be headed by two Project Software Managers, one focused on the ground software and the other focused on the flight software. The Software Managers are assigned the primary responsibility for ensuring that the Project software development management objectives are met. A Software Operations Working Group (SOWG) assists in the coordination and direction of software development activities. The SOWG is composed of representatives from Goddard Space Flight Center, the Project Science Team, and the providers' software organizations.

#### **4.2.1 Software Management Responsibilities**

The Project will identify two Software Managers (SM) who are responsible for ensuring that software being acquired by the Project meets requirements and is delivered on schedule and within budget. One of the Software Managers focuses on ground software and reports to the NGST System Engineer and the other focuses on the flight software and reports to the Project Manager. The SMs' responsibilities include the following:

- review and approve providers' software management plans.
- ensure, at the conclusion of each life cycle phase, that re-estimations of software size, effort, and schedule are made and analyzed.
- serve as chairperson at the software portion of all major reviews. The SM will ensure that all review items are resolved.
- ensure that provider software is delivered in accordance with the Project Master Schedule.

- co-chair the Software Operations Working Group.
- provide technical direction to software providers and support contractors especially on issues which potentially have long-term effects on system schedule and cost.
- ensure commonality of software, hardware and tools where possible to minimize life-cycle cost to the project. Providers will utilize project resources when necessary to ensure the common usage of hardware, software and tools. Providers will identify in their SMP those items they expect will be provided by the Project.
- ensure that all software assurance and configuration management functions are performed.

#### 4.2.2 Software Operations Working Group

The Software Operations Working Group (SOWG) is chaired by the Software Manager and has, among its members, representatives from each of the providers' software organizations. The Software Manager coordinates the activities of the SOWG to accomplish the Software Management responsibilities listed in Section 4.2.1 above.

The SOWG also performs the following functions:

- Coordinates with Project's Systems Engineering groups
- Influences commonality across all software systems. Specific software teams work technical details and interfaces
- Promotes effective systems and software engineering (related to system architecture, policies and practices) for benefit of design, development, I&T and sustaining engineering
- Coordinates SW requirements and interface definitions
- Establishes goals, concepts and approaches for strategic and tactical problems and issues related to the software development activities
- Tracks baseline products to ensure consistency with build and integration plans
- Resolves SW end-to-end operational issues

#### 4.2.3 Provider Organizational Requirements

Software providers shall designate software management representatives who will serve as points-of-contact for the SOWG. In general, the providers' software representatives will serve as members of the SOWG and will act as liaison between the SOWG and the providers' software organizations to help coordinate the providers' software development activities in accordance with the Projects goals and objectives.

### 4.3 Planning, Monitoring and Controlling Mechanisms

Adequate methods and processes for planning, monitoring, assessing progress and status, and taking corrective action are essential to fulfilling the Project's objectives for satisfying technical requirements, meeting delivery schedules, and using resources effectively.

#### 4.3.1 Planning and Estimating

The Project, GSFC Code 5XX, the Prime Contractors, Science Team, and the Science Team contractors, shall establish cost estimates and a Master Schedule based on the WBS described

below. Progress and status will be tracked against the WBS structure. Proprietary information will be segregated to prevent inappropriate disclosure.

#### *4.3.1.1 Work Breakdown Structure*

Each software provider shall develop a Work Breakdown Structure (WBS) for use in planning and costing its work. Each provider shall coordinate this WBS with appropriate project personnel to ensure proper integration into the project WBS.

The Work Breakdown Structure for the Project is as follows:

1. Project Management
2. Science
3. Observatory Systems Engineering
4. Integration and Test (I&T)
5. Component Technology Development
6. Component 1
7. Spacecraft Module 1
8. Spacecraft Module 1
9. Flight software
10. Launch Vehicle
11. Ground Segment and Operations

Ground software development and integration activities will be allocated to lower-level WBS structures under the WBS item related to the associated system segment. In developing lower-level WBS structures, the following items should be considered:

- Segment System/Subsystem/Component structure – Significant system elements are typically allocated to higher WBS elements than the activities that implement them so that resources can be traced to tangible products.
- Planning and Management - includes the development and administration of all software planning information involving cost, schedule and risk, management and control board meetings, and management reviews.
- Acquisition/Purchase - includes the development of all procurement and purchase documents, source selection activities, and acceptance testing of commercial off-the-shelf (COTS) software.
- Logistics and Administrative Support - activities required to acquire and distribute software development supplies and materials; activities required to maintain and operate equipment and facilities used by the development staff. Includes transportation of equipment and staff travel.

- Formulation - includes all conceptual engineering; encompassing development of "throw away" prototypes, supporting analysis, requirements definition, and the development of architecture design documentation.
- Implementation - activities associated with the production and control of computer code including modification or enhancement of inherited, purchased or government furnished software and operation of software development repositories.
- Operation and Maintenance - activities required to effectively utilize software that has been established as an operational baseline, to enhance the software in response to changing requirements or user concerns, and to correct operational deficiencies.
- Performance Assurance - with the exception of Verification and Validation includes all software assurance functions described in Section 6.
- Verification and Validation - includes the tracing, testing and analysis activities described in Section 6.3 of this plan.

#### *4.3.1.2 Activity Identification*

Estimation will begin with identification of activities and elements of cost as the definition of Work Packages of a granularity appropriate to the phase of the project. Consideration should be given to the life cycle model described in Section 5.1.1

Providers shall identify requirements for new, modified and commercial or furnished software components. Software units having common application, if any, shall be specifically identified. Estimates shall be provided of estimated total lines of code broken into appropriate categories.

Revisions to the code size estimates shall be developed by providers at major reviews such as the System Requirements Review (SRR), the Preliminary Design Review (PDR), and the Critical Design Review (CDR).

#### *4.3.1.3 Resource Requirements*

Estimates of resource requirements shall be compiled by each provider and aggregated by the Software Manager to provide budget information to the Project and create a basis for monitoring activities. Estimates will take into account the following resource categories:

- Labor
- Equipment
- Materials, Facilities, and other Resources
- Management Reserves

The provider's SMP shall describe how development staff hours will be allocated to each WBS element for each life cycle phase. Other resources such as travel, equipment, materials, and facilities shall also be listed in the provider's SMP. The actual resource data will be maintained separately.

#### *4.3.1.4 Assumptions, Dependencies, and Constraints*

Providers shall determine existence of dependencies among work packages and on external factors. Also, providers shall identify any assumptions that have been made regarding

circumstances or conditions enabling the performance of any of the work packages, and any constraints or conditions that must be met to accomplish the objectives of any work package.

These assumptions, constraints, and dependencies, along with estimates and schedules, will be reviewed by the SOWG, which will contribute to broader understanding of the intricacies of the project and the relationships among the providers.

#### *4.3.1.5 Budget and Resource Allocation*

Providers shall identify specific resources to be allocated to work packages to meet the resource requirements identified as described in Section 4.3.1.3.

#### *4.3.1.6 Schedules*

The Project will maintain a Master Schedule showing the dates of key events. Providers shall produce schedules, consistent with the Project Master Schedule, showing planned start and end dates for the activities defined by the work packages. These schedules will incorporate the dependencies and resource allocations described in Sections 4.3.1.4 and 4.3.1.5. Use of an automated tool is strongly recommended. Resulting schedules will be discussed in the SOWG, presented at major reviews, and will be posted on the Project or provider web site for access via the intranet.

### **4.3.2 Progress Assessment**

Each provider shall routinely prepare and forward monthly software management and status reports to the Project. These reports shall provide numerical assessments of the status of each activity underway with regard to schedule, cost, staffing and risk. These reports will be aggregated by the Software Manager to present an encompassing summary of the Project software development efforts.

### **4.3.3 Reviews**

This section addresses two types of reviews: (1) formal, project-level reviews, before a review board, and (2) less formal, on-project, software-only reviews. Formal external reviews will be conducted at significant points along the development life cycle, and pertinent information on software systems will be presented at all these reviews. Refer to Project schedules and major contract Statements of Work for details of specific formal project-level reviews.

Each formal review shall include a Software Management Review. The Software Management Review will address the current status of the provider's software accomplishments. The review will present technical plans and accomplishments as well as planned and actual expended staff-hours, available resources and schedules. The software provider shall provide updated estimates of cost- and schedule-to-complete for Project review and approval. The accepted estimates will be used for assessment during the next phase of development.

Software-only reviews at the subsystem level will be held as appropriate throughout the duration of the project. Since development activities for the various software subsystems and successive builds of these subsystems will occur at different times, reviews will be scheduled so as to support development schedules. These reviews shall include Software Requirements Reviews (SRRs), Preliminary Design Reviews (PDRs), Critical Design Reviews (CDRs) and code

walkthroughs. Affected organizations will be encouraged to send representatives to these reviews to ensure that the components are being developed as expected by members of the software community represented by the SOWG.

Software issues surfaced at all reviews will be worked and tracked by the SOWG until they are resolved. Reports of the status of these issues and actions will be maintained and disseminated at regular intervals.

#### **4.3.4 Resource and Schedule Management**

Primary responsibility for resource and schedule management lies with the providers' management personnel. Any indications, resulting from progress assessment activities or reviews, that schedules will not be met or that costs will be exceeded, will be reviewed by the Software Manager and discussed with the provider's software management representative. The Software Manager will assess the impacts of possible deviations and present these findings to the appropriate member of the Project management staff. The Software Manager will then present any significant issues to the SOWG for discussion so that other providers can offer suggestions and solutions and make any necessary adjustments.

#### **4.4 Risk Management**

The Project is conducting a risk assessment and evaluation process. The SOWG will conduct a similar process specifically oriented to risk elements related to the software development efforts. A Risk Management Plan will be produced to define the risk management terminology and procedures. The Risk Management Plan will be reviewed with the SOWG to ensure that all software providers have an opportunity voice their preferences. Risk management plans and processes may be required from Software Providers in accordance with their respective contracts. This process will depend on inputs to be generated by software providers and will consist of the following elements:

- Identification of risks
- Risk Analysis
- Risk Mitigation
- Risk Monitoring

The process will address the following classes of risk:

- Technical Risks
- Safety Risks
- Security Risks
- Resource Risks
- Schedule Risks
- Cost Risks

#### **4.5 Configuration Management**

The Project will define an approach to configuration management involving centralized, automated, on-line processes. It is desirable to have a single software CM system, however, providers may utilize proprietary systems if that is shown to be most cost-effective. In any case,

all CM databases shall be available, at least for read access, to the Project, to Systems Engineering teams, and to Software Development and Integration and Test Teams either government or contractor.

The Software Manager, assisted as necessary by the SOWG, will provide support to the Project in the following areas with regard to software development:

- Develop the Project's Software Configuration Management Plan.
- Establish the Project's Software Configuration Management (CM) system.
- Serve as a member of the Project level CCB.
- Establish and maintain the Project's Software Change Request (CR) tracking data base.
- Establish and maintain the Project's Software Problem Reporting and Resolution System.
- Review CRs for software classification changes.
- Manage the SCM library and thereby control the use and revision of official copies of baseline components.
- Produce and distribute periodic Software CR data base and individual product CR status reports.
- Support Project functional and physical configuration audits (FCA & PCA) of providers.
- Review providers' Software Configuration Management Plans.

The SOWG and the software providers will work together to structure a configuration management process for the items listed below, and the providers shall also address these in their Software Management Plans or in separate Configuration Management Plans.

- Configuration Identification
- Configuration Change Control
- Configuration Status Accounting
- Configuration Authentication

#### **4.6 Documentation Requirements**

For each software segment and configuration item, the provider will identify, in the corresponding Software Management or Product Plan, the appropriate documentation to be prepared. These will include:

- Requirements Specifications
- Interface Requirement Documents
- Interface Control Documents
- Design Documents
- Integration and Test Plans and Reports
- Build and Release Plans, Test Plans and Reports
- Verification and Validation Plans
- Product Assurance Plans
- Delivery and Transition Plans
- Sustaining Engineering and Operations Activity Plans
- Users Guides

- Operations and Maintenance Guides

#### **4.7 Deviation Procedures**

Deviations can be proposed during major reviews or requested between reviews. Generally, deviations will be approved if circumstances indicate that it is advantageous to the project and any additional risk incurred from doing so is minimal and justified by the advantage gained. Deviations may not be employed to avoid proper treatment of critical software or to circumvent product assurance measures. Typical deviations might include such things as use of a non-standard higher-order language or use of a non-traditional development or testing approach.

Providers, when preparing their SMPs, shall describe any major or encompassing deviations they anticipate so that these may be reviewed early on.

## 5 TECHNICAL APPROACH

The Project software engineering process will employ systematic technical approaches with appropriate adherence to accepted guidelines and standards for both development and sustaining engineering phases of the mission. Providers are required to present and distribute data products and specifications in a manner consistent with the project information infrastructure. Providers will also be required to collect data on technical performance that supports measurement and improvement of the effectiveness of technical processes.

### 5.1 Software Engineering Process

The Project Software Engineering Process will be governed by a life-cycle model intended to provide a framework that:

- ensures that a thorough and systematic process is employed
- ensures that adequate opportunities for interaction are provided for all stakeholders who can beneficially contribute
- supports the generation of products useful to the development and maintenance processes

Providers are required to document in their SMPs the processes, methods, development environments and tools, standards and guidelines that they will employ at every stage of the life cycle.

#### 5.1.1 Life Cycle Model

Project Software Development will generally follow the flow of a traditional life cycle with the following principal phases and activities:

##### Formulation Phase

- Operations Concept Definition
- Requirements Definition
- Requirements Analysis
- Architecture Design

##### Implementation Phase

- Software Preliminary Design
- Software Detailed Design
- Software Code and Unit Test
- Software Integration and Test

##### Operations and Maintenance Phase

- Deployment
- Acceptance Testing
- Transition
- Operations
- Sustaining Engineering

The intention is to apply this life cycle flexibly so that emphasis is on performing useful work that contributes useful products. For cases where a particular activity would not add value, it should be modified or eliminated. Some of these activities will overlap and, in some cases, run

currently. Also, different components may be in different phases and activities at a given time as specified in Build Release Plans.

### 5.1.2 Formulation Phase

The Formulation Phase involves defining the system or component to be built in terms of a description of its purpose, how it will operate in the context within which it will be used, what functions it will perform, what requirements and constraints it must satisfy, and how it will be organized at the highest level. It is expected that there will be a lot of interaction between the various activities that contribute to the formulation process.

Other related activities that will be taking place during the formulation phase include planning the development and integration effort and developing initial estimates of cost and schedule.

#### 5.1.2.1 *Operations Concept Definition*

The operations concept is a description of the purpose of the system to be built and descriptions of how it will operate in terms that users of the system will understand and relate to. In the case of systems that are used by other systems, the operations concept can be useful to developers of the user systems.

The purpose of the operations concept is to ensure that the right system is being built and that users will find the completed system useful. The operations concept is the foundation for validation activities.

A science mission operations concept is being developed by the Science Team. From this top-level operations concept, the operational characteristics of segments and components can be defined. Operations concept information should encompass all modes and functions including contingency cases. Techniques for presenting operations concept information include scenarios, operations timelines, use cases, and user interface prototypes.

#### 5.1.2.2 *Requirements Definition*

Requirements are clear, unambiguous, testable, quantitative, well organized, hierarchically structured statements about what the system will do including the functions it will perform, how it interacts with other entities, and how well (fast, reliably, etc.) the system will perform them. Detailed requirements shall be identified as functional, performance, and interface requirements. Functional and performance requirements are based on and derived from operations concept information. Interface requirements may depend on information from the software architecture design process (Section 5.1.2.5) to define the entities between which interfaces must be specified. Requirements for interface between segments or major systems shall be expressed in Interface Requirements Documents (IRDs).

The hierarchical structure of the requirements manifests as a set of mission requirements (see Reference 3), traceable to the Science Goals, and sets of more detailed or lower-level requirements that can be traced to the mission requirements. There may be as many as five or six levels of requirements culminating in requirements for software objects at the component level. Traceability shall be maintained between higher- and lower-level requirements. Typical designations for the levels of requirements are as follows.

Level 0            Science Goals

Level 1	Mission Requirements
Level 2	System Requirements
Level 3	Segment Requirements
Level 4	Element Requirements
Level 5	Subsystem Requirements
Level 6	Component or Unit Requirements

The highest level requirements that are purely software requirements will probably appear at level 3 or 4. Requirements should be maintained in one or more data bases viewable by all members of the project to facilitate the construction of verification matrices for use in tracing requirements to design elements and test processes. Requirements must be reviewed, discussed and approved in requirements review meetings led by the SOWG.

#### *5.1.2.3 Requirements Analysis*

Requirements analysis comprises any activities intended to ferret out the implications of requirements to provide a strong foundation for design. This may take the form of logical analysis of information requirements, event analysis, data flow analysis, or state transition analysis. This type of analysis may result in lower-level requirements, definition of algorithms, or other information specifying the nature of the processing to be done.

Requirements analysis is also used to develop hardware resource requirements for support of development and processing operations. Hardware resource requirements include such things as processor and storage capacities, and network and device throughput requirements.

Requirements analysis is the process for ensuring that the requirements are properly understood so that design may proceed. The need for requirements analysis depends on the complexity of the requirements and the detail of the operations concept information available. Software providers will indicate in their SMPs areas where requirements analysis will likely be required. The SOWG may designate areas requiring further analysis, for example, during reviews of operations concept information and requirements.

The Project will review the software requirements at a formal System Requirements Review (SRR). A Requirements baseline will be established after the completion of the SRR.

#### *5.1.2.4 Architecture Design*

Software architecture design involves identification of the principal or top-level Computer Software Configuration Items (CSCI) of a system and the allocation of all software requirements to these configuration items. Software providers will group software requirements into logical sets such that each set maps to a CSCI. As a general rule, a CSCI is established for a separable piece of the software system that can be designed, implemented, and operated independently. Other criteria that may affect the decision to designate a software entity as a CSCI are:

- critical to overall performance
- highly complex, incorporates new technologies, involves high risk, or has stringent performance or safety requirements
- higher than usual expected modification rates

- encompasses all of a specific domain of functionality
- will be installed on a separate computer platform distinct from other parts of the system
- encapsulates interfaces with other software items that currently exist or are provided by other organizations
- some part of the software is planned to be reused

Software architecture design may also involve preliminary identification of off-the-shelf components, preliminary estimates of computational resource requirements, and the allocation of software to hardware platforms.

The Project will review the software architectural design at a formal system Preliminary Design Review (PDR). A Software Allocated Baseline, will be established after the completion of the PDR. The Allocated Baseline will contain the architecture design of the system and documents showing how the requirements are allocated to the design. It shall also contain all the updated documents from the Requirements baseline, along with the architectural design specification materials.

### 5.1.3 Implementation Phase

The Implementation Phase includes preliminary design, detailed design, coding and unit testing, integration of all components, and testing up to the system and acceptance levels. This Software Management Plan assumes that object-oriented methods and programming languages will be used since this generally represents the state-of-the-practice at this time.

Providers are strongly encouraged to use automated tools to facilitate the development process and to follow accepted standards for presentation of design information and formatting of implementation products.

Providers are encouraged to utilize “off-the-shelf” products to the greatest extent practical as a means for reducing development costs.

#### 5.1.3.1 Software Preliminary Design

Software Preliminary Design refines the software architecture design to the point where all software components and interactions between components are identified down to the unit level. Also, principal data structures and external files are identified. Typical design and presentation approaches include object diagrams, class diagrams, entity relationship diagrams, and data base schemata. It is highly desirable that design products be viewable or downloadable from the intranet.

Final decisions are made regarding which components will be purchased “off-the-shelf” and which components will be developed, and what programming languages and development environments will be used. Opportunities for reuse of common software elements are identified.

The Software Preliminary Design is reviewed, at a minimum, in a software-only, on-project review open to members of the SOWG and a product assurance representative. Reviews will be scheduled sufficiently in advance so that SOWG members and product assurance representatives can plan to attend. Products of the Software Preliminary Design activity shall be maintained in a Software Development Folder (see Section 5.1.5.1).

#### *5.1.3.2 Software Detailed Design*

Software Detailed Design specifies the details of the computer methods and programs, data structures, and unit test approaches. Software Detailed Design products are reviewed in a peer walkthrough scheduled sufficiently in advance so that a product assurance representative and, possibly, SOWG members can plan to attend. At these walkthroughs, in depth discussion of the coding and testing techniques and approaches may be required so they must be attended by the personnel who will write and test the code, typically the same personnel who developed the detailed design. Products of the Software Detailed Design activity shall be maintained in a Software Development Folder (see Section 5.1.5.1).

The Project will review the software detailed design at a formal Critical Design Review (CDR). A Software Detailed Design Baseline will be established after the completion of the CDR. The Software Detailed Design Baseline will contain the computer methods and programs, data structures, and unit test approaches. It shall also contain all the updated documents from the Requirements baseline and Software Allocated baseline, along with the detailed design materials.

#### *5.1.3.3 Software Code and Unit Test*

Coding and unit testing are performed in accordance with the Detailed Design products. Code walkthroughs are conducted to verify that code conforms to the coding standard, that the logic and approaches are consistent with the detailed design, and that implementation details have been constructed in a reasonable and favorable manner. Unit tests are performed to verify the logic and proper execution of the code. Coded and tested units are transferred to a configuration managed software repository in preparation for Software Integration and Test.

#### *5.1.3.4 Software Integration and Test*

Planning for Integration and Test begins during the Formulation Phase with the definition of the levels of integration and the sequence of builds and releases. The system may be delivered in several builds or blocks, each of which may have several different releases with increasing levels of capability.

Test planning begins after requirements are allocated to architectural components. A general method for verifying each requirement is identified for inclusion in the test plan. Groups of requirements are assembled based on similarity of function and commonality of test approach. A test procedure is identified to verify requirements in each group.

Each provider shall prepare an Integration and Test Plan that describes the sequence and levels of integration from the component level up to the level of the delivered system to be provided by that provider. The I&T Plan shall describe the facilities and approaches to be employed in this process including test tools, simulators, actual flight hardware, engineering test units, and components of interfacing systems. Automated test methods shall be used wherever practicable to facilitate regression testing and re-testing.

Subsequent, higher levels of integration and testing, including end-to-end system testing, verification and validation, and certification are discussed under Product Assurance in Section 6 of this document.

#### 5.1.4 Operations and Maintenance Phase

The operations and maintenance phase encompasses activities involving the deployment of completed systems, the acceptance of the systems by user organizations, and the transition of those systems to supporting the activities of users and operators who are preparing for or conducting mission operations including software system maintenance. This portion of the SMP will be revised as the Project develops. There may be differences among the various segments and components

##### *5.1.4.1 Deployment*

Deployment involves the distribution of copies of the software to sites where it will be operated and installation of the software on the designated computer equipment including any initial configuration needed to bring the newly installed software into fully operating status.

##### *5.1.4.2 Acceptance Testing*

Acceptance testing is the final testing activity whereby the entity receiving the products participates in a formal test of the system using an agreed upon set of test procedures that demonstrate the proper functioning of the system, at the level of user interaction. The Acceptance Test Procedures shall be designed to demonstrate the operational readiness of all functional and performance capabilities of the system as called for in the requirements. Acceptance testing may be conducted as part of the Verification and Validation activity (see Section 6.3).

##### *5.1.4.3 Transition*

Transition involves the training of personnel and development of personnel procedures for operating the system to fulfill its intended purpose.

Planning for Deployment and Transition typically begins at a very high level around PDR and becomes successively more refined throughout the Implementation Phase. Software personnel may be required to support activities related to facility preparation, and hardware installation and checkout prior to software deployment.

##### *5.1.4.4 Operations*

Simulated operations, a logical follow-on from the validation process, will begin sufficiently before launch so that adequate training can be conducted for support of both the deployment phase and normal operations.

Operations procedures will be developed by personnel responsible for conduct of science and spacecraft operations. These procedures are likely to be based on procedures developed by Spacecraft and Science Instrument developers who will most likely remain involved, at least during some initial or transition period, until such time as operations are running smoothly.

Software providers will be responsible for development of User Guide or Operations Manual type information, probably housed in online repositories for electronic access from operations work stations.

#### 5.1.4.5 Sustaining Engineering

Sustaining engineering is required for all flight and ground software, and for the software and tools related to sustaining engineering environments. The sustaining engineering environments will most likely be identical to or close variants of the development and validation environments used for software implementation and for data and procedure development.

Sustaining engineering consumes a significant portion of the project life-cycle cost. Consequently, considerations for this should begin as early as the architectural design phase. Particular regard should be given to required facilities and personnel, and to the impact of architectural decisions that enhance the use of common elements thereby reducing maintenance complexity.

#### 5.1.5 Other Software Engineering Considerations

This section discusses software engineering topics that do not fit conveniently into the discussions above related to life cycle phases.

##### 5.1.5.1 Software Categorization and Classification Policy

Each software subsystem or component shall be classified into one of three categories depending on how critical that subsystem or component is to the safety and operation of the mission. The three categories are defined in Table 5-1. Once classified, a Software Change Request is required to change the classification. Requests to change classification will require review by the software manager. Providers shall describe in their SMPs how their software development practices will differ among the three categories.

**Table 5-1 Software Classification**

CATEGORY	NAME	DESCRIPTION
1	Mission Critical	Software that directly impacts the safety of the spacecraft (such as safing system software); requires highly stringent development, testing and verification methods.
2	Operations Essential	Software that needs to operate properly to support operations or sustaining engineering; requires “industry best practices” for software development, testing and verification.
3	Temporary	Software that does not need to be maintained because it is used only once or over a short interval of time and will not be needed subsequently. Any products used in operations, produced by temporary software, must be verified in accordance with Category 2 practices.

#### *5.1.5.2 Software Development Folders*

Software Development Folders (SDFs) are a means for tracking and documenting the development of software components. SDFs contribute to a systematic process for development of software units, support information gathering and process improvement, and protect against losses due to unanticipated absences of development personnel. Information is placed in SDFs during Software Preliminary Design, Software Detailed Design, Software Code and Unit Test, and possibly during Integration and Test.

Each provider shall establish and document procedures for creating and maintaining SDFs in their SMP. SDFs shall meet the following requirements:

- The provider shall document the development of each computer software unit (CSU), computer software component (CSC), and CSCI using software development folders (SDFs). The provider shall establish a separate SDF for each such software item or logically related group of items.
- The provider shall establish the folders within one month after the completion of the preliminary design review and shall maintain the SDFs until the delivery of the final product and completion of the contract.
- The SDFs shall be made available for Project review upon request. SDFs may be generated, maintained, and controlled by automated means. To reduce duplication, SDFs should not contain information provided in other documents or SDFs.
- The SDFs shall include (directly or by reference) the following information:
  - Design requirements
  - Design considerations and constraints.
  - Design documentation and data
  - Schedule and status information
  - Source code.
  - Test requirements and responsibilities.
  - Test cases, procedures, and results.
  - Configuration control activities and change reports to include traceability of the requirement for change, the authorizing authority, and the changes made to software/documentation.
  - Results of walkthroughs, reviews, and inspections, with findings and recommendations and actions taken.

#### *5.1.5.3 Data Generation and Management*

In addition to software components involving computer code, the project will require significant data components that will play a variety of roles in ground and flight systems. Data sets will be classified by how they are used operationally. This includes how the data are created, how the data are stored and accessed, who provides the data, who uses the data, how long the data are retained, how often and under what circumstances the data are modified, relationships to other data, and how the data are validated and certified.

For example, data sets may be system related in which case they might be treated much the same as software components; they may be spacecraft related requiring occasional update by spacecraft system engineers; they may be used for test purposes requiring early generation and

subsequent update as system requirements evolve; or, they may be routine operations oriented constituting a regular flow of information through the system.

Software providers shall develop basic information for all data sets identified during preliminary design so that refinements to operations concept information can be made and operations planning can proceed, including the specification of the necessary data management systems. This information will be updated at PDR and refined at CDR so that changes can be reflected in operations planning and procedures documentation.

The required information for each type of data shall include:

- Name or type of data
- Description of the data
- How the data are created and by whom
- How the data are stored
- How the data are accessed and by whom
- Volume of the data
- Data retention and archive requirements
- How often and under what circumstances the data are modified
- Relationships to other data
- Validation and certification requirements

#### ***5.1.5.4 Problem Reporting***

Each provider shall operate an automated problem reporting system that can accept problem reports from developers, testers, and users of the systems being developed. The problem reporting system shall be able to track and report the status of all problems that have been recorded and selectively report categories of reports based on the recorded fields. It is recommended that one large encompassing problem reporting system be constructed to serve the entire project. This problem reporting system could be combined, as an additional capability, with the change control and reporting system referred to in Section 4.5 that tracks the progress and status of change requests.

## **5.2 Data Product and Specification Structure**

The data products and specifications to be prepared in the course of development activities are tabulated below with explanatory comments.

**Table 5-2 Project Software Specifications and Data Products**

<b>DATA PRODUCT</b>	<b>COMMENTS</b>
Software Management or Product Plan (includes or rollout documents for) Implementation Plan CM Plan Risk Management Plan Product Assurance Plan	Initial draft prepared during Formulation Phase, revised for PDR and CDR.
Interface Requirements Document	Preliminary IRDs between major segments are being developed by prime contractors with SOWG support. They will contain sections to address the software aspects of the interfaces. These include: <ul style="list-style-type: none"> <li>• Ground System to Observatory IRD</li> <li>• Component to Component IRDs</li> </ul> Other IRDs may be defined as appropriate between lower-level software components. Final IRDs are generally required by PDR for the components involved.
Interface Control Document	Preliminary ICDs are prepared at the start of detailed design. Final ICDs are generally required by CDR prior to commencement of fabrication.
System Requirements Specification	Preliminary SRS is required by software System Requirements Review. Final SRS is due at software PDR
Design Document	Software design documentation is prepared during preliminary and detailed software design activities. It may be produced largely with automated tools. Design documentation is finalized after the software PDR or CDR for that component.
Build and Release Plans	Build and Release Planning begins during architectural design and continues through CDR.
Build and Release Test Plans and Reports	Preparation of Test Plans begins during definition of the entity (build or release) to be tested. Reports of the results of testing activities are prepared within 10 days of the completion of the test activities.
Integration and Test Plan and Reports	Integration and Test Planning begins during architectural design for system levels and continues through CDR for subsystem and component levels. Reports of the results of testing activities are prepared within 10 days of the completion of the test activities.
Verification and Validation Plans and Reports	Draft V&V Plans should be in place by the end of the Formulation Phase and should be finalized early in the Implementation Phase.
Delivery and Transition Plan	Delivery and Transition Planning should begin no later than CDR and be completed by Acceptance Testing.
Sustaining Engineering and Operations Activities Plan	Sustaining Engineering Planning should begin no later than CDR and be completed before any transfer of maintenance responsibilities begins.
Users Guide	User documentation should begin no later than CDR and be completed by System Acceptance Testing (see section 6.4).
Operations Manual	Operations documentation should begin no later than CDR and be completed by System Acceptance Testing (see section 6.4). Should cover software and data components.
Maintenance Manual	Maintenance documentation should begin no later than CDR and be completed by System Acceptance Testing (see section 6.4). Should cover software and data components.
Data Component	Data components will be treated in accordance with procedures for the type of data. Generally, persistent data will be configuration managed as either a software system component or an operations data component.

Software Component	Software components will be placed under configuration management, in approved repositories, upon submission for integration after completion of unit testing.
--------------------	--

### 5.3 Technical Performance Measures

The Project will use metrics as management and quality indicators. To support this use, each provider shall establish and implement a software metrics program that will enhance their capabilities to manage and direct the software development process and facilitate the growth of product quality.

Software metric data shall be collected that support the quantitative evaluation and analysis of trends for the entire life cycle development process and the products that it generates. The specific metrics to be collected will be defined by PDR. Examples of metrics that may be useful include:

- Requirements established/modified/deleted
- Software change requests opened/closed/remaining open/cumulative
- Estimated source lines of code
- Design/code complexity index
- Percent memory, CPU, and I/O utilization
- Source code growth rate
- Detected code error rates
- Problem reports opened/closed/remaining open/cumulative
- Effort data (staffing profile)
- Development CPU time usage and trends
- Number of audits, inspections, reviews, walk-throughs , etc.
- Development activity status

The collection, reporting and analysis of metrics shall be automated to the fullest extent practicable and shall be performed at appropriate intervals or nominally on a monthly basis. Metrics shall be provided to the Project both as raw data and in graphical form.

## **6 PRODUCT ASSURANCE**

This section identifies processes for ensuring that all NGST software-related products meet the requirements and standards that have been identified as essential to achieving NGST Software Development Project goals.

### **6.1 Quality Assurance**

Quality Assurance for this project will be conducted in accordance with the GSFC Quality Management System which consists of all the policies, processes and procedures that contribute to meeting GSFC's ISO 9001 quality goals as expressed in Reference 5, The GSFC Quality Manual, GPG 8730.3C. For Flight Software, the GSFC Information Systems Center, Code 582, will direct the Quality Assurance process.

Quality Assurance involves establishing standards for the preparation of data products, monitoring the products to ensure that they conform to the standards, and recommending corrections when they do not. Standards are discussed further in Section 6.2.

Each software provider shall describe their approach to software quality assurance. This description may be either part of their Software Management Plan or in a separate Software Quality Assurance Plan.

### **6.2 Applicable Standards**

Appropriate standards should be used as guidelines for development of software, data and documentation products. In many cases, however, standards for presentation, formatting and style are imposed by automated tools that are selected for a variety of features and benefits. Consequently, it is not prudent to rigidly specify that certain standards be followed because that could preclude adoption of tools that provide significant advantage. It is important that sound standards are identified and that the intent of the selected standards are met. Consequently, the selection of tools should consider the methodologies employed and the features of the output products that the standards are intended to engender.

The designation and utilization of standards is particularly recommended for the following items:

- Requirements
- Design
- Coding standards
  - Formatting Conventions
  - Usage Rules
  - Portability – avoid non-standard usage
  - In-code documentation
    - Prolog – Arguments, I/O, Exceptions, Change History, Version
    - Algorithm Description
    - Variable Description
    - PDL
- Unit Test
- Documentation
  - Planning documents

➤ User and Operator Manuals

Each software provider will identify in their Software Management Plan the standards that will be used in their software development processes. Sections 2.2 and 2.3 suggest various standards that may be employed.

### **6.3 Verification and Validation**

Verification ensures consistency from product to product along the development path and involves constructing verification matrices showing the allocation of requirements to both software components down to the module level and to test processes to ensure that all functional and performance requirements are met. Verification also involves independent review and or testing of all development products at a system level. Validation consists of testing and utilizing all development products in a real or simulated operational environment, in the manner intended for users and operators, to ensure that the system meets the needs of the user and operator community. End-to-end system and acceptance test functions, or portions of them, may be conducted as part of the verification and validation activities.

Validation shall be performed, to the greatest extent practicable, by persons independent of the development activities.

Each provider shall describe in their Software Management Plan the specific processes and procedures for performing verification at the component and system level. Verification at the mission level and system validation will be described in a Project Integration and Test Plan.

### **6.4 Safety Assurance**

Safety Assurance requires that all systems and facilities are designed and maintained so as to cause no physical harm to workers, users, or observers. For software, this will primarily involve ensuring that the physical environments where software development and testing are performed are free of hazards and conform to standard workplace safety standards and guidelines. Software commanding of spacecraft mechanisms in testing environments may also be a consideration.

### **6.5 Security**

Security Assurance involves all necessary measures, both technical and procedural, to ensure that (1) all processing systems supporting the conduct of Project flight operations cannot be accessed from outside the community intended for its use; and (2) data products generated by these systems, as well as the systems themselves, cannot be altered by unauthorized persons either maliciously or unintentionally. This will include measures to ensure computer security, information security and physical security.

### **6.6 Certification**

Certification of software and data components for use in the direction or execution of spacecraft flight operations involves a methodical accounting process for ensuring that every requirement has been tested, every change has been verified, and every data item has been checked. Procedures for accomplishing this for Project software are TBD.

## 7 ABBREVIATIONS AND ACRONYMS

CDR	Critical Design Review
CM	Configuration Management
CMP	Configuration Management Plans
COTS	Commercial Off-the-Shelf
CPU	Central Processing Unit
CR	Change Request
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
ESC	Electrical Systems Center
FCA	Functional Configuration Audit
GSFC	Goddard Space Flight Center
I&T	Integration and Test
I/O	Input/Output
ICD	Interface Requirements Document
IEEE	Institute of Electrical and Electronics Engineers
IRD	Interface Requirements Document
NASA	National Aeronautics and Space Administration
PCA	Physical Configuration Audit
PDL	Program Design Language
PDR	Preliminary Design Review
SDF	Software Development Folders
SI	Science Instrument
SM	Software Manager
SMP	Software Management Plan
SOWG	Software Operations Working Group
V&V	Verification and Validation
WBS	Work Breakdown Structure
WWW	World Wide Web